TILING MODELS FOR SPATIAL DECOMPOSITION IN AMTRAN

John Compton and Christopher Clouse

Lawrence Livermore National Laboratory

Joint Russian-American Five-Laboratory Conference on Computational Mathematics/Physics

Vienna, 2005

UCRL-PRES-212696

This work was performed under the auspices of the United States Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract Number W-7405-ENG-48.

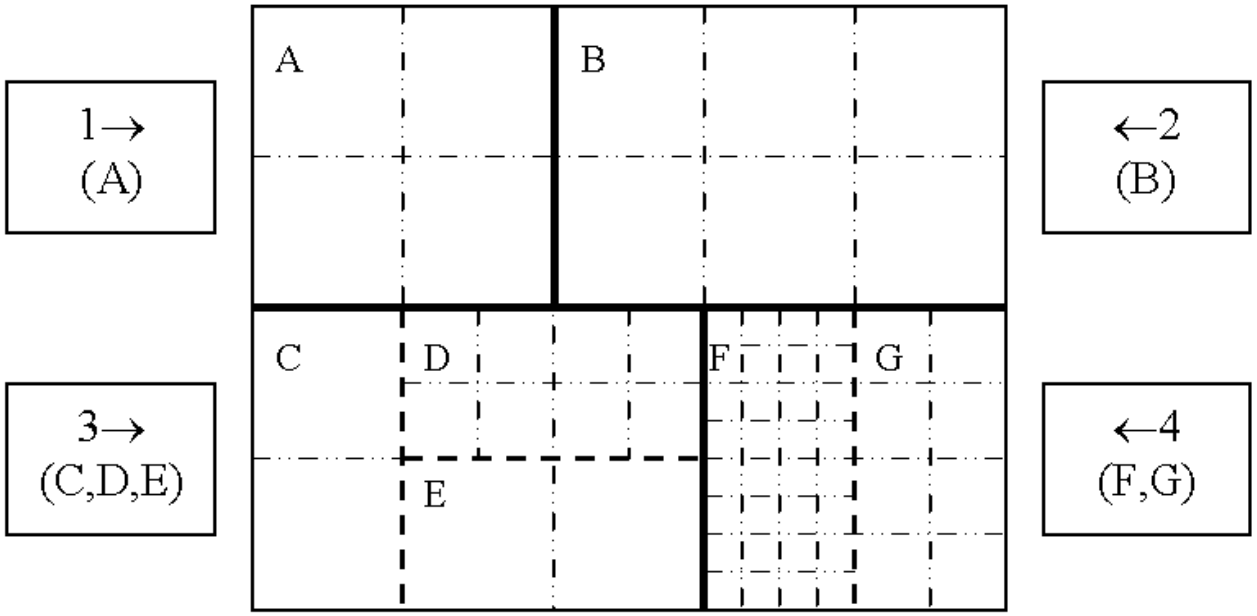
Overview of AMTRAN code

- AMTRAN is a deterministic (Sn) neutron transport code
- Computation is performed on a collection of Cartesian finite element grids
- Grids are at various levels of refinement based upon the local mean free path of the neutrons
- The time-independent transport equation is $\vec{\Omega} \bullet \vec{\nabla}\Psi_g^m + \sigma_{tot}\Psi_g^m = S(\psi)$
- It is solved using a set of coupled single-energy group transport equations, each of which can be solved in parallel (e.g., 32 separate energy groups)
- It is further discretized into angles, each of which may be likewise computed independently (typically 3 to 16 angles)
- In our implementation the energy groups are distributed among the processors and the angles are distributed among the threads
- The biggest remaining potential source of parallelism is in spatial domain decomposition (in theory unlimited, currently implemented to 512 domains)

Definitions

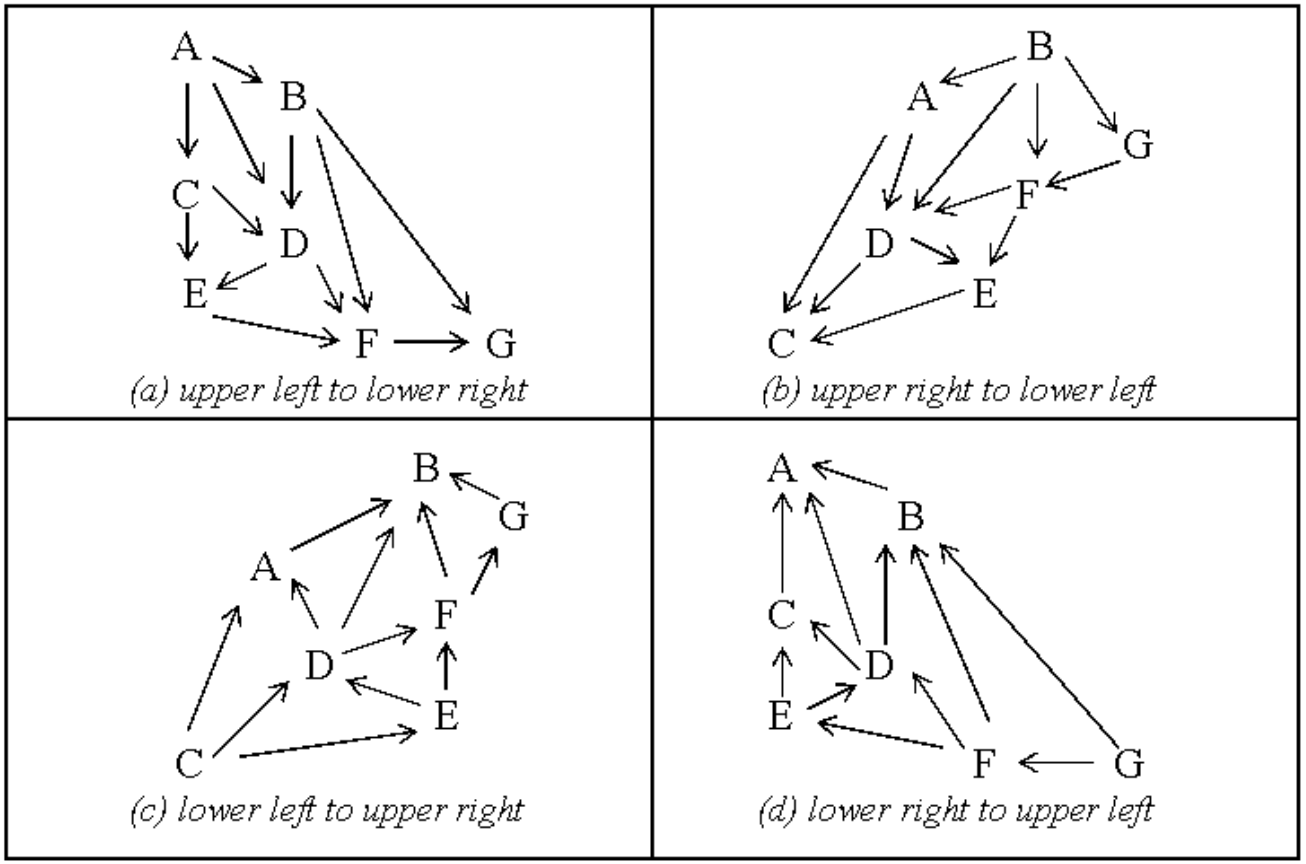
- Grid: a rectangular prism (brick) subdivided into zones according to a uniform Cartesian mesh
- Domain: a collection of grids making up a rectangular prism (brick) representing a portion of the physical space being modeled
- Master: an equivalence class of domains which reside on a single processor or in parallel on a group of processors
- Sweep: a grid computation associated with a direction (+1 or -1 in each of the Cartesian axes)
- Sub-iteration: the smallest unit of computation (consisting of a collection of sweeps) that can be performed without the exchange of messages among processors
- Iteration: a collection of sub-iterations (punctuated by message passing) in which every domain is swept from every possible direction
- Cycle: a collection of iterations which converges to an answer

Simple example of a two-dimensional domain decomposition



Processor	Master	Domains	Grids	Energy groups
1	1	1,4	A,F,G	a,b
2		1,4	A,F,G	c,d
3	2	2	B	a,b
4		2	B	c,d
5	3	3	C,D,E	a,b
6		3	C,D,E	c,d

Example of sweep dependencies (precedence relations) for seven grids in four domains



- Upper left and lower right boxes show inverse relationship
- Similarly for lower left and upper right boxes

Simplifying assumptions

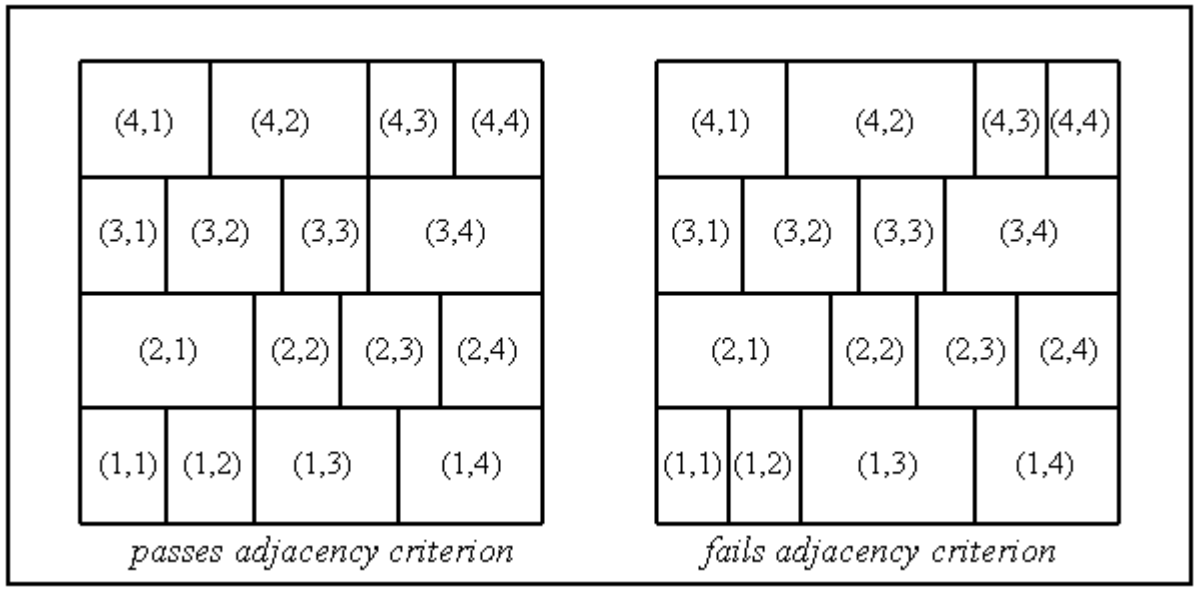
- The goal is to obtain a set of canonical configurations with known properties
- Treat all grids in a domain as a single featureless computational unit
- Perform domain splits in a strict sequence to achieve powers of two uniformly on each axis
- Any pair of processors either operates on the same domains (possibly discontinuous parts of the physical space) or they have no domains in common, i.e., belong to equivalence classes according to their master
- The number of domains per master is uniform and a power of two, determined by the particular configuration of splits
- The determination of which sweeps will be performed in a sub-iteration are determined by a script in order to segregate work into equal time packets

Order and degree of partitioning according to number of domains

Number of Dimensions	Number of Domains	Number of Regions Partitioned along Axis 1	Axis 2	Axis 3
2	4	2	2	
2	8	2	4	
2	16	4	4	
3	8	2	2	2
3	16	2	2	4
3	32	2	4	4
3	64	4	4	4
3	128	8	4	4
3	256	8	8	4
3	512	8	8	8

- Binary decomposition determined by an approximate weighting scheme (accurate to ±20%)
- Choice of 1st, 2nd, 3rd axis among Cartesian axes (x,y,z) is based upon geometrical considerations and presence of symmetry planes
- Order of computations closely tied to order of decomposition

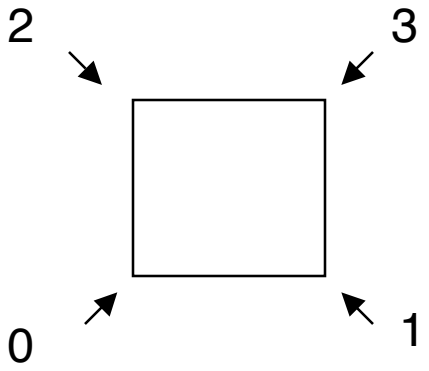
Illustration of domain adjacency criterion



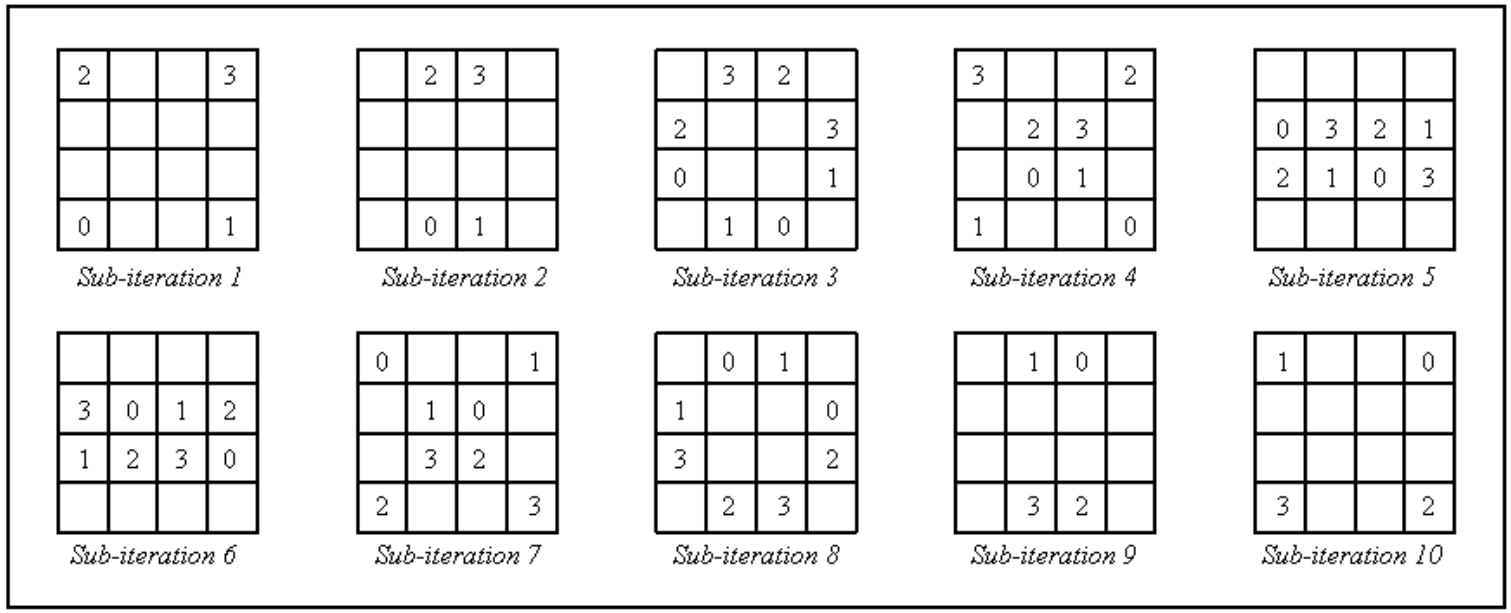
- In order to pass adjacency criterion, domains sharing a common border must have coordinates differing by no more than one in any direction
- In the example above, the initial vertical decomposition guarantees the criterion for the first coordinate
- In the example on the right domains (4,2) and (3,4) share a border, hence violating the criterion in the second coordinate

Numerical representation of sweep directions

Sweep Direction	Number of Dimensions	Interpretation
0	2	lower left to upper right
1	2	lower right to upper left
2	2	upper left to lower right
3	2	upper right to lower left
0	3	lower left front to upper right back
1	3	lower right front to upper left back
2	3	lower left back to upper right front
3	3	lower right back to upper left front
4	3	upper left front to lower right back
5	3	upper right front to lower left back
6	3	upper left back to lower right front
7	3	upper right back to lower left front



Sweep pattern for 16 domains in two dimensions



- Vertical lines do not necessarily align, but adjacency criterion met
- Blank domains represent absence of computation
- Each domain is swept in each of four directions
- Overall efficiency rate is only 40% (4 sweeps per 10 sub-iterations)

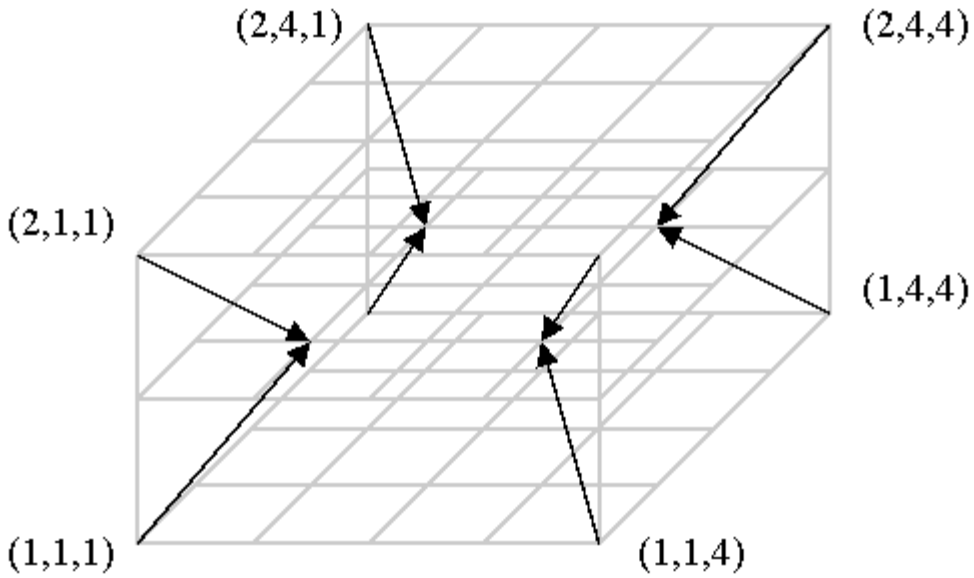
Sweep directions for a 4x4 set of domains

Master	Domain	Sub-iteration (without domain overloading)									
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
		Sweep Directions									
1	(1,1)	0			1			2			3
2	(1,2)		0	1					2	3	
3	(1,3)		1	0					3	2	
4	(1,4)	1			0			3			2
5	(2,1)			0		2	1		3		
6	(2,2)				0	1	2	3			
7	(2,3)				1	0	3	2			
8	(2,4)			1		3	0		2		
9	(3,1)			2		0	3		1		
10	(3,2)				2	3	0	1			
11	(3,3)				3	2	1	0			
12	(3,4)			3		1	2		0		
13	(4,1)	2			3			0			1
14	(4,2)		2	3					0	1	
15	(4,3)		3	2					1	0	
16	(4,4)	3			2			1			0

Master	Domains	Sub-iteration (with domain overloading)									
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
		Sweep Directions									
1	(1,1),**(3,1)**	0		**2**	1	**0**	**3**	2	**1**		3
2	(1,2),**(3,2)**		0	1	**2**	**3**	**0**	**1**	2	3	
3	(1,3),**(3,3)**		1	0	**3**	**2**	**1**	**0**	3	2	
4	(1,4),**(3,4)**	1		**3**	0	**1**	**2**	3	**0**		2
5	(2,1),**(4,1)**	**2**		0	**3**	2	1	**0**	3		**1**
6	(2,2),**(4,2)**		**2**	**3**	0	1	2	3	**0**	**1**	
7	(2,3),**(4,3)**		**3**	**2**	1	0	3	2	**1**	**0**	
8	(2,4),**(4,4)**	**3**		1	**2**	3	0	**1**	2		**0**

- Left side is for 16 masters (and domains) with overall efficiency of 40%
- Right side represents superposition of upper and lower halves of left side, giving 8 masters (still 16 domains) with efficiency of 80%
- Left side can be viewed as a discontinuous tile for tiling the plane at ±8 units vertically and ±8 units horizontally (as on the right)

Schematic view of 32 domains in a 4x4x2 configuration



- Arrows represent sweeps for the first sub-iteration
- Some corner domains are numbered according to their coordinates
- The order of the coordinates represents the order of binary decomposition according to axis

Tabular representation of sweeps for 32 domains in a 4x4x2 decomposition

Master	Domains	Sub-iteration																	
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
		Sweep Directions																	
1	(1,1,1),**(1,3,1)**	0		**2**	1	**0**	**3**	2	**1**	*4*	3	***6***	*5*	***4***	*7*	6	***5***		*7*
2	(1,1,2),**(1,3,2)**		0	1	**2**	**3**	**0**	**1**	2	3	*4*	*5*	***6***	*7*	***4***	***5***	*6*	*7*	
3	(1,1,3),**(1,3,3)**		1	0	**3**	**2**	**1**	**0**	3	2	*5*	*4*	*7*	***6***	***5***	***4***	*7*	*6*	
4	(1,1,4),**(1,3,4)**	1		**3**	0	**1**	**2**	3	**0**	*5*	2	*7*	*4*	***5***	***6***	*7*	***4***		*6*
5	(1,2,1),**(1,4,1)**	**2**		0	**3**	2	1	**0**	3	***6***	**1**	*4*	*7*	*6*	*5*	***4***	*7*		***5***
6	(1,2,2),**(1,4,2)**		**2**	**3**	0	1	2	3	**0**	**1**	***6***	*7*	*4*	*5*	*6*	*7*	***4***	***5***	
7	(1,2,3),**(1,4,3)**		**3**	**2**	1	0	3	2	**1**	**0**	*7*	***6***	*5*	*4*	*7*	*6*	***5***	***4***	
8	(1,2,4),**(1,4,4)**	**3**		1	**2**	3	0	**1**	2	*7*	**0**	*5*	***6***	*7*	***4***	***5***	*6*		***4***
9	(2,1,1),**(2,3,1)**	4		**6**	5	**4**	7	6	5	0	7	**2**	1	0	**3**	2	**1**		3
10	(2,1,2),**(2,3,2)**		4	5	**6**	7	**4**	**5**	6	7	0	1	**2**	**3**	**0**	**1**	2	3	
11	(2,1,3),**(2,3,3)**		5	4	7	**6**	**5**	**4**	7	6	1	0	**3**	**2**	**1**	**0**	3	2	
12	(2,1,4),**(2,3,4)**	5		7	4	**5**	**6**	7	**4**	1	**6**	**3**	0	**1**	**2**	3	**0**		2
13	(2,2,1),**(2,4,1)**	**6**		4	7	6	5	**4**	7	**2**	**5**	0	**3**	2	1	**0**	3		**1**
14	(2,2,2),**(2,4,2)**		**6**	7	4	5	6	7	**4**	**5**	**2**	**3**	0	1	2	3	**0**	**1**	
15	(2,2,3),**(2,4,3)**		7	**6**	5	4	7	6	**5**	**4**	**3**	**2**	1	0	3	2	**1**	**0**	
16	(2,2,4),**(2,4,4)**	7		5	**6**	7	4	**5**	6	3	**4**	1	**2**	3	0	**1**	2		**0**

- 16 masters, 2 domains per master, 18 sub-iterations
- 2 copies of tile horizontally, ½+½+½+½ vertically (total of 4)
- Theoretical efficiency = 16/18 (89%)

Tabular representation of sweeps for 64 domains in a 4x4x4 decomposition

Masters	Domains	Sub-iteration																				
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
		Sweep Directions																					
1	(1,1,1),**(3,1,1)**	0			1	4		2	5	0	3	6	1	4	7	2	5		3	6			7
2	(1,1,2),**(3,1,2)**		0	1			4	5	2	3	0	1	6	7	4	5	2	3			6	7	
3	(1,1,3),**(3,1,3)**		1	0			5	4	3	2	1	0	7	6	5	4	3	2			7	6	
4	(1,1,4),**(3,1,4)**	1			0	5		3	4	1	2	7	0	5	6	3	4		2	7			6
5	(1,2,1),**(3,2,1)**			0		2	1	4	3	6	5	0	7	2	1	4	3	6	5		7		
6	(1,2,2),**(3,2,2)**				0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7			
7	(1,2,3),**(3,2,3)**				1	0	3	2	5	4	7	6	1	0	3	2	5	4	7	6			
8	(1,2,4),**(3,2,4)**			1		3	0	5	2	7	4	1	6	3	0	5	2	7	4		6		
9	(1,3,1),**(3,3,1)**			2		0	3	6	1	4	7	2	5	0	3	6	1	4	7		5		
10	(1,3,2),**(3,3,2)**				2	3	0	1	6	7	4	5	2	3	0	1	6	7	4	5			
11	(1,3,3),**(3,3,3)**				3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4			
12	(1,3,4),**(3,3,4)**			3		1	2	7	0	5	6	3	4	1	2	7	0	5	6		4		
13	(1,4,1),**(3,4,1)**	2			3	6		0	7	2	1	4	3	6	5	0	7		1	4			5
14	(1,4,2),**(3,4,2)**		2	3			6	7	0	1	2	3	4	5	6	7	0	1			4	5	
15	(1,4,3),**(3,4,3)**		3	2			7	6	1	0	3	2	5	4	7	6	1	0			5	4	
16	(1,4,4),**(3,4,4)**	3			2	7		1	6	3	0	5	2	7	4	1	6		0	5			4
17	(2,1,1),**(4,1,1)**	4			5	0		6	1	4	7	2	5	0	3	6	1		7	2			3
18	(2,1,2),**(4,1,2)**		4	5			0	1	6	7	4	5	2	3	0	1	6	7			2	3	
19	(2,1,3),**(4,1,3)**		5	4			1	0	7	6	5	4	3	2	1	0	7	6			3	2	
20	(2,1,4),**(4,1,4)**	5			4	1		7	0	5	6	3	4	1	2	7	0		6	3			2
21	(2,2,1),**(4,2,1)**			4		6	5	0	7	2	1	4	3	6	5	0	7	2			3		
22	(2,2,2),**(4,2,2)**				4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3			
23	(2,2,3),**(4,2,3)**				5	4	7	6	1	0	3	2	5	4	7	6	1	0	3	2			
24	(2,2,4),**(4,2,4)**			5		7	4	1	6	3	0	5	2	7	4	1	6	3	0		2		
25	(2,3,1),**(4,3,1)**			6		4	7	2	5	0	3	6	1	4	7	2	5	0	3		1		
26	(2,3,2),**(4,3,2)**				6	7	4	5	2	3	0	1	6	7	4	5	2	3	0	1			
27	(2,3,3),**(4,3,3)**				7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
28	(2,3,4),**(4,3,4)**			7		5	6	3	4	1	2	7	0	5	6	3	4	1	2		0		
29	(2,4,1),**(4,4,1)**	6			7	2		4	3	6	5	0	7	2	1	4	3		5	0			1
30	(2,4,2),**(4,4,2)**		6	7			2	3	4	5	6	7	0	1	2	3	4	5			0	1	
31	(2,4,3),**(4,4,3)**		7	6			3	2	5	4	7	6	1	0	3	2	5	4			1	0	
32	(2,4,4),**(4,4,4)**	7			6	3		5	2	7	4	1	6	3	0	5	2		4	1			0

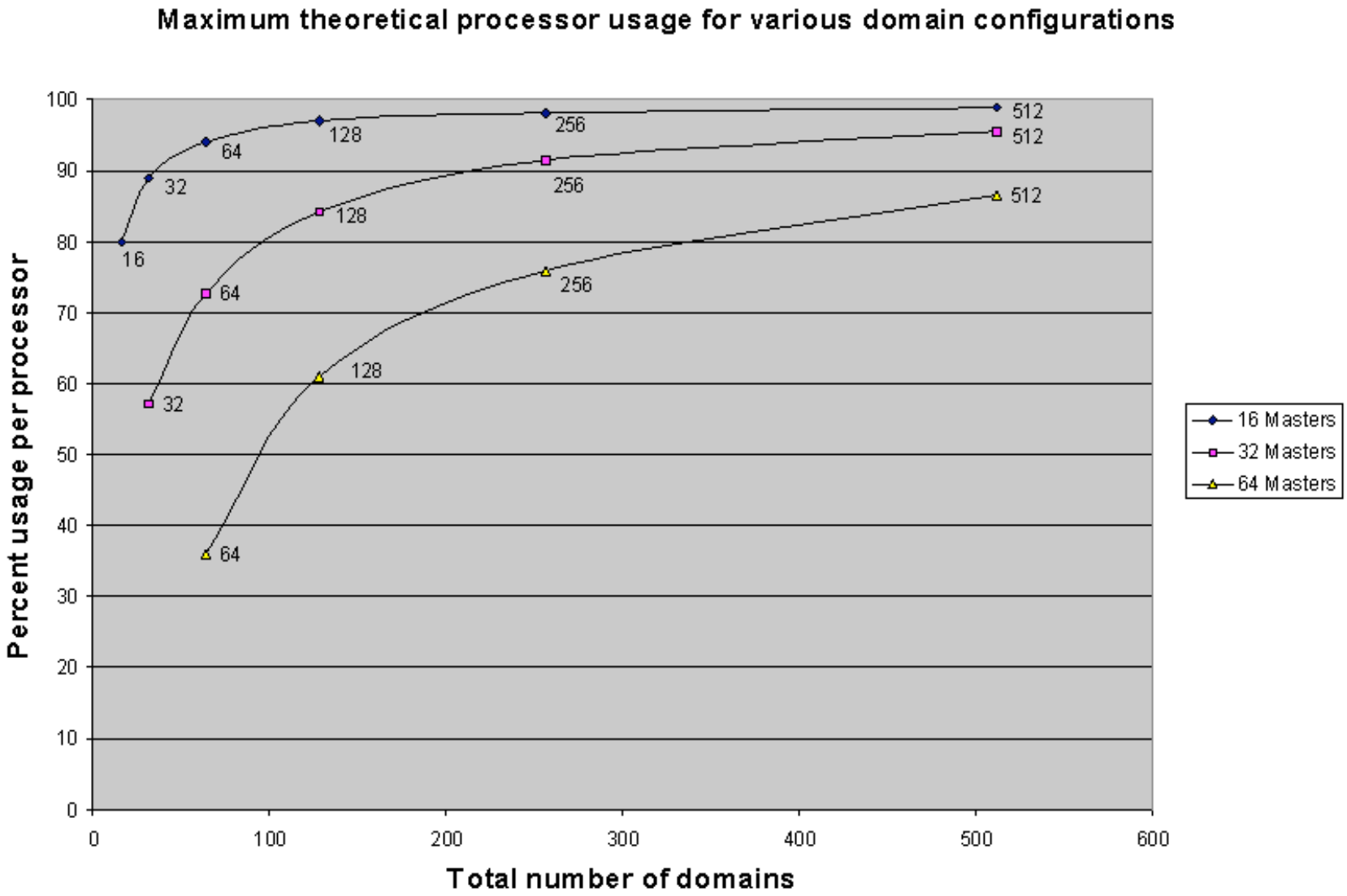
- 32 masters, 2 domains per master, 22 sub-iterations
- 4 copies of tile horizontally, 2 vertically (total of 8)
- Tile replication at 4 units horizontally, 16 units vertically
- Theoretical efficiency = 16/22 (73%)

Theoretical efficiencies for various domain decomposition strategies

Dimensions	Masters	Domains	Domains per Master	Active Sub-iterations	Total Sub-iterations	Percent Efficiency
2	4	4	1	4	4	100
2	8	8	1	4	6	67
2	8	16	2	8	10	80
2	16	16	1	4	10	40
3	8	8	1	8	8	100
3	16	16	1	8	10	80
3	16	32	2	16	18	89
3	16	64	4	32	34	94
3	16	128	8	64	66	97
3	16	256	16	128	130	98
3	16	512	32	256	258	99
3	32	32	1	8	14	57
3	32	64	2	16	22	73
3	32	128	4	32	38	84
3	32	256	8	64	70	91
3	32	512	16	128	134	96
3	64	64	1	8	22	36
3	64	128	2	16	26	61
3	64	256	4	32	42	76
3	64	512	8	64	74	86
3	128	512	4	32	50	64
3	128	1024	8	64	82	78
3	128	2048	16	128	146	88
3	128	4096	32	256	274	93

- All but the largest configurations have been implemented
- “Theoretical” means perfect load balance and no communications overhead

Computational efficiencies

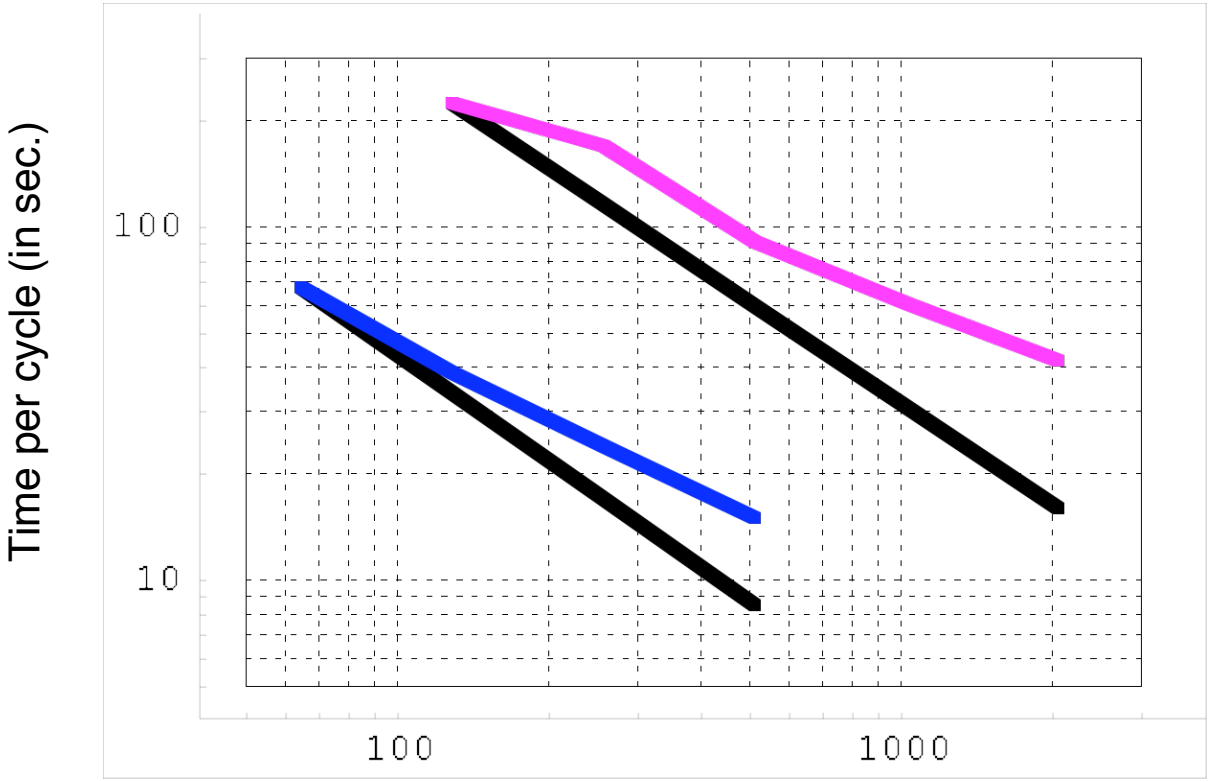


Timing comparison of theoretical efficiencies with domain overloading

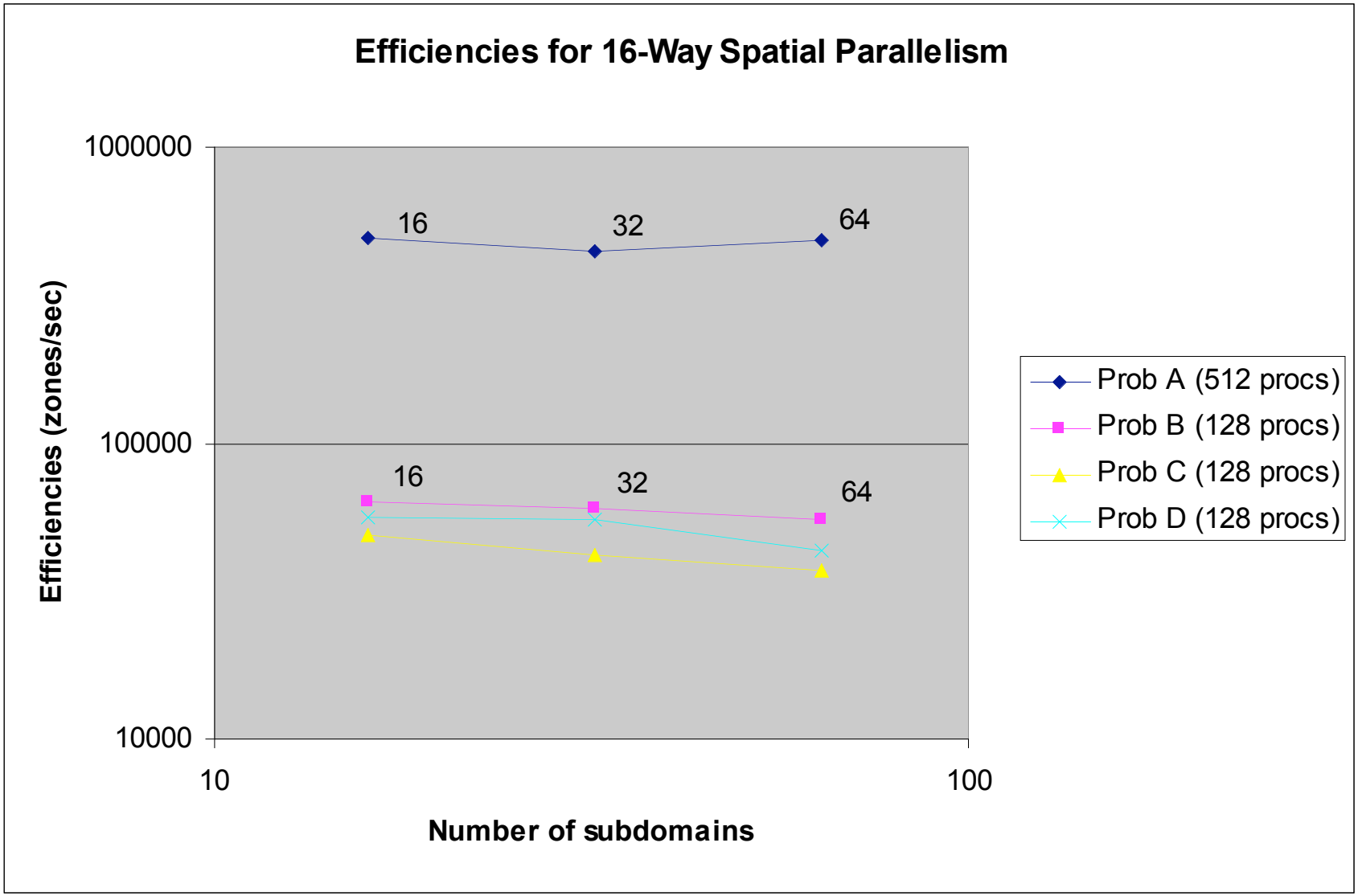
Number of masters	Number of domains	Time (seconds)	Theoretical efficiency	Efficiency × time
16	16	1440	.80	1150
16	32	1330	.89	1184
16	64	1272	.94	1196

- Problem has s8 quadrature with 32 energy groups, fourfold symmetry
- Corresponds to timing of ~1200 seconds for 100% efficiency
- Efficiency × time is nearly constant, as expected
- With lower symmetry such results are more difficult to achieve
- In general, larger configurations (with more domains) require corresponding finer meshes (to allow partitions to be accurately placed for best load balance)
- Other timing studies generally bear out the results shown here
- Configurations with up to 512 domains have been successfully tested

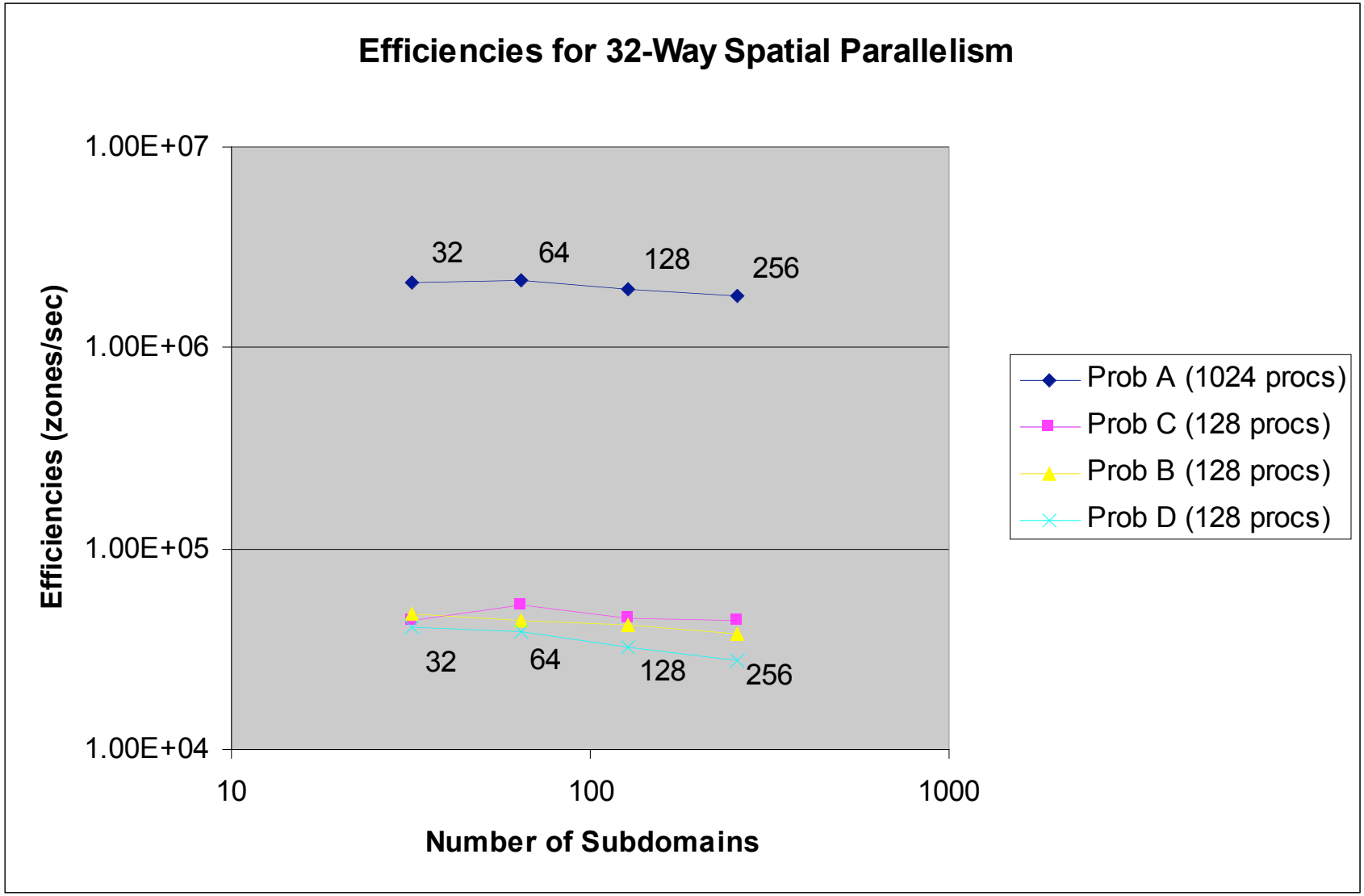
Efficiency loss in distributing a constant problem to larger numbers of processors



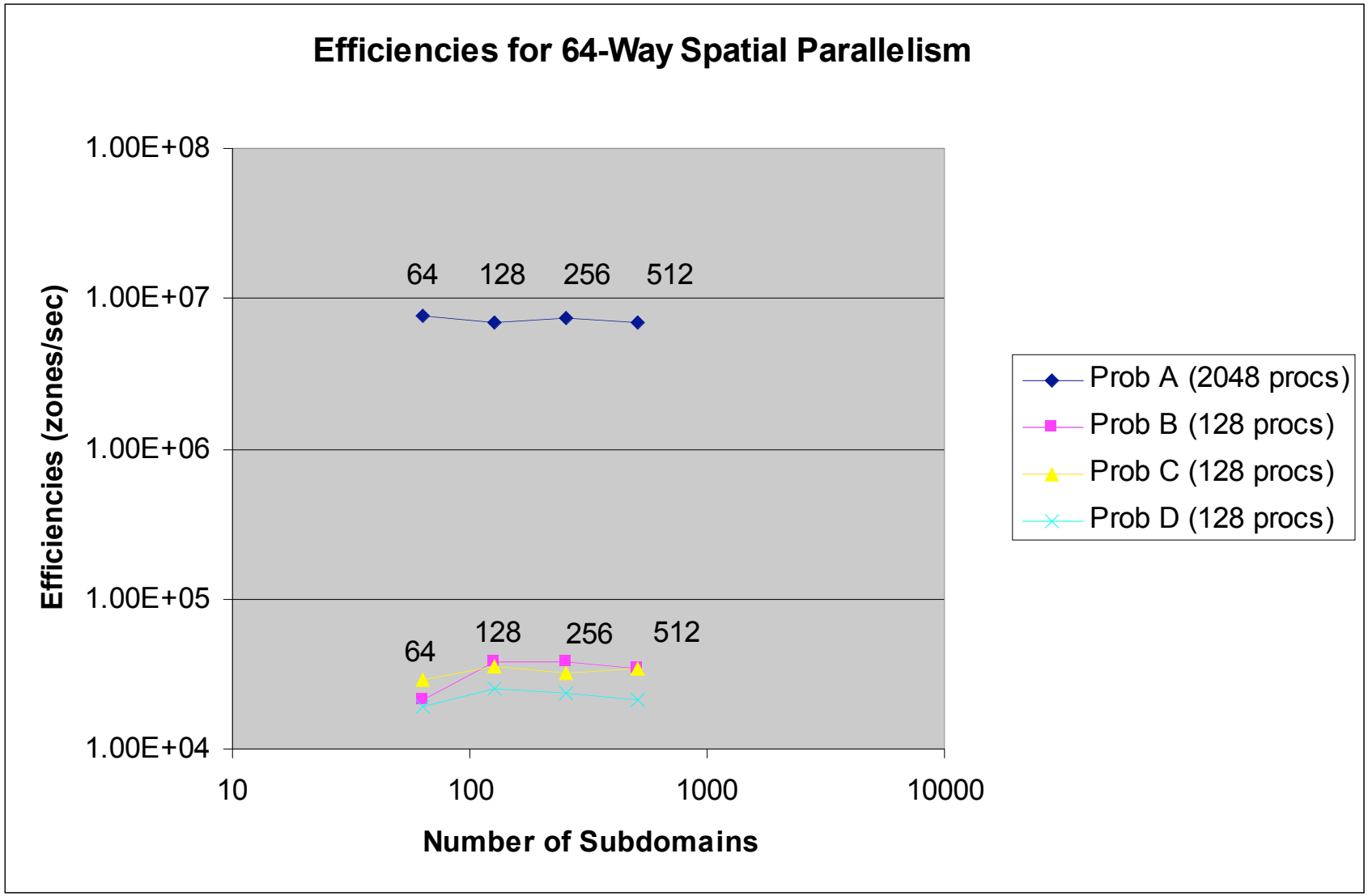
Effect of domain overloading in the case of 16-way spatial parallelism



Effect of domain overloading in the case of 32-way spatial parallelism



Effect of domain overloading in the case of 64-way spatial parallelism



Other considerations and conclusions

- Dynamic load rebalancing (after the initial decomposition) has been tested and found to be effective for small numbers of domains
- Enforcing the adjacency criterion at the expense of perfect load balancing is effective, otherwise scripting of sweeps is impossible and global timing conflicts result
- For large numbers of domains it is important that there be no “outliers” on the high side (in terms of load balance) since they dominate the computation
- The automated load-balancing algorithm was tested against load balancing by hand to verify its effectiveness
- The new scheme has enabled computations to be performed to a much higher degree of parallelism than previously